Border Basis for Polynomial System Solving and Optimization

Philippe Trébuchet, Bernard Mourrain, Marta Abril Bucero

Border basis for polynomial system solving and optimization

Ph. Trébuchet[1] & B. Mourrain[2] & M. Abril Bucero[2]

[1] ANSSI, Paris, France
Philippe.Trebuchet@lip6.fr
[2] Inria Sophia Antipolis Méditerranée, AROMATH, France
Bernard.Mourrain@inria.fr, Marta.Abril_Bucero@inria.fr

Abstract. We describe the software package BORDERBASIX dedicated to the computation of border bases and the solutions of polynomial equations. We present the main ingredients of the border basis algorithm and the other methods implemented in this package: numerical solutions from multiplication matrices, real radical computation, polynomial optimization. The implementation parameterized by the coefficient type and the choice function provides a versatile family of tools for polynomial computation with modular arithmetic, floating point arithmetic or rational arithmetic. It relies on linear algebra solvers for dense and sparse matrices for these various types of coefficients. A connection with SDP solvers has been integrated for the combination of relaxation approaches with border basis computation. Extensive benchmarks on typical polynomial systems are reported, which show the very good performance of the tool.

1 Border basis algorithms

In this section, we briefly describe the border basis algorithms and the algebraic solvers available in the package BORDERBASIX . Let $R = \mathbb{K}[x_1, \ldots, x_n]$ be the ring of polynomials in the variables $x_1, \ldots, x_n$ with coefficients in a field $\mathbb{K}$. Let $f_1, \ldots, f_m$ be the equations to be solved and $I = (f_1, \ldots, f_n)$ the ideal of R generated by these equations. The algebraic approach implemented in this package to solve the set of equations $\{f_1, \ldots, f_m\}$ proceeds in two steps:

a) Compute the quotient algebra structure $\mathcal{A} = R/I$ represented by a (monomial) basis and the operators of multiplication by the variables.
b) Compute the roots of the system from the operators of multiplication by the variables, when $\dim \mathcal{A} < \infty$.

The main algorithm of the package BORDERBASIX is the computation of border bases which provides the algebra structure of $\mathcal{A}$.

A border basis is defined with respect to a set B of monomials, connected to 1 (if $m \in B$ either $m = 1$ or $\exists i_0 \in [1, n]$ and $m' \in B$ such that $m = x_{i_0} m'$). Let $B^+ := B \cup x_1 B \cup \ldots \cup x_n B$ and $\partial B = B^+ \setminus B$. The computation of a border basis goes through the construction of a family F of polynomials of the form

$$f_\alpha = \mathbf{x}^\alpha - \sum_{\beta \in B} c_{\alpha,\beta} \mathbf{x}^\beta$$

which are in $\langle B^+ \rangle$ with only one term denoted $\gamma(f_\alpha) = \mathbf{x}^\alpha$ in ∂B and the other monomials of its support in B. A family of polynomials of this form is called a *rewriting* family. The family is *graded* if $\deg(\gamma(f)) = \deg(f)$ for all $f \in F$.

Let $\mathcal{M}$ be the set of monomials in the variables $x_1, \ldots, x_n$. For $S \subset R$, $t \in \mathbb{N}$, S_t is the set of elements of S of degree $\leq t$. We denote by $\langle S \,|\, t \rangle$ the vector space spanned by the elements sm such that $\deg(sm) \leq t$ for $s \in S$ and $m \in \mathcal{M}$.

A rewriting family F is said to be *complete* in degree t if it is graded and satisfies $(\partial B)_t \subseteq \gamma(F)$; that is, each monomial $m \in \partial B$ of degree at most t is the leading monomial of some (necessarily unique) $f \in F$.

Given a rewriting family F which is complete in degree t, we define recursively the projection $\pi_{F,B}$ on B along F in the following way: $\forall m \in \mathcal{M}$,

- if $m \in B_t$, then $\pi_{F,B}(m) = m$,
- if $m \in (\partial B)_t$ $(= (B^{[1]} \setminus B^{[0]})_t)$, then $\pi_{F,B}(m) = m - f$, where f is the (unique) polynomial in F for which $\gamma(f) = m$,
- if $m \in (B^{[k]} \setminus B^{[k-1]})_t$ for some integer $k \geq 2$, write $m = x_{i_0} m'$, where $m' \in B^{[k-1]}$ and $i_0 \in [1, n]$ is the smallest possible variable index for which such a decomposition exists, then $\pi_{F,B}(m) = \pi_{F,B}(x_{i_0}\, \pi_{F,B}(m'))$.

The map $\pi_{F,B}$ extends by linearity to a linear map from $\mathbb{K}[\mathbf{x}]_t$ onto $\langle B \rangle_t$. By construction, $f = \gamma(f) - \pi_{F,B}(\gamma(f))$ and $\pi_{F,B}(f) = 0$ for all $f \in F_t$. The next theorem shows that, under some natural commutativity condition, the map $\pi_{F,B}$ coincides with the linear projection from $\mathbb{K}[\mathbf{x}]_t$ onto $\langle B \rangle_t$ along the vector space $\langle F \,|\, t \rangle$ (see [16]):

Theorem 1. *Assume that B is connected to 1 and let F be a rewriting family for B, complete in degree $t \in \mathbb{N}$. Suppose that, for all $m \in \mathcal{M}_{t-2}$,*

$$\pi_{F,B}(x_i\, \pi_{F,B}(x_j\, m)) = \pi_{F,B}(x_j\, \pi_{F,B}(x_i\, m)) \text{ for all } i, j \in [1, n]. \tag{1}$$

Then $\pi_{F,B}$ coincides with the linear projection of $\mathbb{K}[\mathbf{x}]_t$ on $\langle B \rangle_t$ along the vector space $\langle F \,|\, t \rangle$; that is, $\mathbb{K}[\mathbf{x}]_t = \langle B \rangle_t \oplus \langle F \,|\, t \rangle$.

The commutation polynomials or C-polynomials are the polynomials:

$$\pi_{F,B}(x_i\, \pi_{F,B}(x_j\, m)) - \pi_{F,B}(x_j\, \pi_{F,B}(x_i\, m))$$

for $m \in B$, $i, j \in [1, n]$ such that $x_i m \in \partial B$ or $x_j m \in \partial B$.

1.1 Border basis computation

The border basis algorithm computes a rewriting family F which satisfies the relation (1) for any degree t. It proceeds incrementally degree by degree with a candidate monomial set B for the basis of $\mathcal{A}$ and the rewriting family F for B at a given degree t. At each degree, the non-zero polynomials deduced from the relations (1) are added to F.

In the main loop of the algorithm, the following operations are performed:

1. **prolongation**: determine the new elements of B in degree $t + 1$ and the elements $\tilde{F}$ of F^+ which are in $\langle B^+ \rangle$;

2. **matrix construction**: compute the coefficient matrix M of $\tilde{F}$ with respect to B^+;
3. **linear reduction**: compute a (sparse) LU decomposition of the matrix M and update the rewriting family $\tilde{F}$;
4. **commutation reduction**: reduce the C-polynomials with respect to $\tilde{F}$ and update B, F and t;

This loop is iterated until a complete rewriting family for B which satisfies (1) is obtained in the case of a zero-dimensional ideal [16] or until the Gotzmann regularity criterion is satisfied [17]. The computation is controlled by a choice function, which select "leading" monomials for the construction of rewriting families.

1.2 Real radical computation

The border basis algorithm has been extended to compute the real radical of an ideal [14], by integrating in the main loop a new operation:

5. **moment kernel**: compute a linear functional, which is positive on the sum of squares and orthogonal to $\langle F \,|\, t\rangle$, by solving a Semi-Definite Program; compute a basis of the kernel of its moment matrix in degree $\frac{t}{2}$ and add it to F.

1.3 Polynomial optimisation

The border basis algorithm is also extended to compute the minimum of a polynomial $f(\mathbf{x})$ on a basic closed semi-algebraic set S defined by a set of constraints $\mathbf{g}^0 = 0$, $\mathbf{g}^+ \geq 0$ and the points where this minimum is reached (if they exist)[2]. The following operations are inserted in the main loop:

5.1. **optimal moment kernel**: compute a linear functional, which minimizes f on the preordering or quadratic module generated by the set of constraints, by solving a Semi-Definite Program, for moments associated to monomials in $B_{\frac{t}{2}} \cdot B_{\frac{t}{2}}$;

5.2. **flat extension test**: check if the moment sequence satisfies a flat extension property, and compute a basis and the border associated to the moment matrix kernel in degree $\frac{t}{2}$.

1.4 Root finding

In the case of a zero-dimensional ideal, the last step in the resolution process consists in computing the roots from one or several operators of multiplication [5], [11].

The eigenspaces associated to the transposed operator M_1^t of multiplication by the variable x_1 in $\mathcal{A}$ are computed. The first coordinates of the roots are given by the eigenvalues of M_1^t. If the eigenspaces are one-dimensional the other coordinates are deduced. Otherwise the transposed operator of multiplication by x_2 restricted to these eigenspaces is computed as well as its eigenspaces. It determines the second coordinates associated to a given first coordinate. This process is repeated until all coordinates of the roots are determined.

2 Software

The package BORDERBASIX is implemented in C++. It contains classes of multivariate polynomials represented as lists of monomials; classes for the border basis algorithms and for the solution of polynomial systems by eigenvector computation and linear algebra tools for dense and sparse matrices;

All the implementations are parameterized (templated) by the coefficient type. So that it is possible to use several number types for the computation of border bases. The set of number types effectively used in this computation includes modular arithmetic, multi-modular arithmetic, `double`, `long double`, `double double`, extended arithmetic based on GMP such as rational numbers (`mpq`), floating point numbers based on the GMP type `mpf` or on the library `mpfr`.

For linear algebra on dense matrices, a templated version of BLAS and LAPACK [3] libraries has been developed and is available in the `linalg` sub-package of BORDERBASIX. It includes the specialization of some of the arithmetic functions needed to control the precision of the computation. The main functionalities on dense matrices used in this library are Singular Value Decomposition, eigenvalue and eigenvector computation.

For sparse matrices, a templated version of SuperLU [7] is also available, so that it can be used with general arithmetic number type. The main functionality used in this library is the solution of sparse linear systems by a direct method, for the computation of a rewriting family from the coefficient matrix of polynomials in the main loop of the border basis algorithm.

The connection with SDP solvers is developed in two ways. For the solver CSDP, a connection with a templated version has been implemented. The solver MOSEK [18] is also linked directly with the border basis implementation. For the solvers SDPA, SDPA_GMP [9], a file interface is used to describe the SDP problem to be solved and the SDP solver is called in an external process. The result is output in a file and read for the next step of the border basis computation. Since the SDP part is the most expensive part of whole computation, using files is not too penalizing.

The package contains approximatively 250 000 lines[3] of C++ code. It is accessible from `http://www-sop.inria.fr/teams/galaad/software/bbx`. It is also part of the software project MATHEMAGIX (`www.mathemagix.org`).

3 Benchmarks

In this section we analyse the behavior of our software on characteristic inputs:

- Generic zero dimensional systems, because they provide the simplest case, with no degree drop, trivial syzygies and sharp zero dimensional detection;
- Cyclic-n test suite because they lead to very sparse bases and because the computation of the latter involves finding many non trivial syzygies.

[3] counted with cloc

The benchmarks have been performed on an `Intel Corei7-3610QM CPU@ 2.30GHz` with 6Go of DDR3 1600 MHz.

First we compare the basis computation, we emphasize here that once the border basis is computed all the multiplication matrices are available. This is not the case for a Gröbner basis and the computation of just one multiplication matrix can be costly as shown in [12].

We have indicated by `-` test cases that have failed for unexplained reason and by `mem` cases that failed because of lack of memory.

`mac` choice function is the choice function that returns one monomial of highest degree and highest partial degree.

3.1 Katsura-n

The systems Katsura are zero dimensional complete intersection. Resultant theory give a characterization of a basis that is canonical but is not a Gröbner basis. First of all we present timings using modular arithmetic. The reason is that the behavior exposed here is also the one obtained using multi-modular computations. We then present numerical computations and show two different algorithms to recover the roots from the border basis, the first one described in [5], and the second in [11] and [21]. The system katsura-n has 2^n solutions. In the following table, we show *raw* timing using our method and other Gröbner engines.

n	Bbx mac	Bbx grl	Magma [4]	Sing. (std) [6]	Sing. (SlimGB) [6]	Giac [20]	Fgb [8]
7	0.06	0.09	0.018	0.01	0.01	0.05	0.1
8	0.23	0.43	0.14	2	3	0.37	0.5
9	1.05	2.41	0.84	13	35	2.15	2.7
10	5.43	18.23	5.4	100	333	11.6	22.5
11	33.41	127.23	37.26	1043	3408	87.2	172.6
12	240.69	1029.15	602.02	>15000	>15000	715.55	mem
13	1985.35	10432.12	4700.1	-	-	mem	-
14	13121.62	>15000	mem	-	-	-	-

Most of the time difference between *BBx mac* order and *Bbx grl* order is due to the time spent to perform reduction of the C-polynomials, this operation has not been made scalable yet and also would greatly benefit from adaptation of the signature based criterions.

3.2 Cyclic-n

This family of polynomial systems comes from permutation theory. These systems are very far from being a complete intersection and have a complicated first syzygy module making them a standard benchmark case for Gröbner bases computations. It is noticeable here that the Gröbner basis computed for each system is very sparse, i.e. the multiplication matrix is costly to compute from the basis. This partly explains the difference of timings between `bbx` and the other softwares.

n	Bbx mac	Bbx grl	Magma grl [4]	Singular [6]	Giac [20]	Fgb [8]
5	0.12s	0.05s	0.01s	(0.01)	(0.16s)	(0.01s)
6	1.09s	0.18s	0.10(0.01)s	(0.01)	(0.15s)	(6.63s)
7	65.46s	7.24s	12.78(0.28)s	(3)	(0.61s)	(2.6s)

The total time (in seconds) for computing the multiplication matrix by a variable is reported in this table. The time for computing Gröbner bases is given between parentheses. We did not consider higher systems of Cyclic-n which are not zero-dimensional.

What is also striking here is the timing difference between BORDERBASIX and the classical Gröbner engines. The explanation is that for Cyclic-n systems the Gröbner basis is very far from giving at least one multiplication matrix. As shown in [12] in such a case the computation of the multiplication matrix is the bottleneck of the resolution process. For instance it took 12.5 second with Magma to compute the multiplication matrix by x_0, the first variable, for the Cyclic-7 problem.

3.3 Floating Point computation

In this section we present the floating point computation that are available inside BORDERBASIX. We show the time needed and the accuracy of the computed basis (the error is computed from the rational basis).

We use the Katsura-6 system as support benchmark, for it has only 64 solutions that are suitable for double precision treatment. The error estimates and approximations are performed, computing a certification as exposed in [11] and [21].

arith	time basis	time solve	time cert	error on basis	error on sol
MPQ	22.5	–	–	0	–
double	0.058	0.06	0.60	10^{-10}	10
long double	0.069	0.32	3.34	10^{-30}	10^{-14}
$MPF128$	0.13	3.38	17.64	10^{-38}	10^{-36}
$MPF256$	0.25	3.57	56.9	10^{-76}	10^{-75}

We present here the same comparison for katsura-7:

arith	time basis	time solve	time cert	error on basis	error on sol
MPQ	22.5	–	–	0	–
double	0.058	–	–	10^{-10}	–
longdouble	0.68	0.32	3.35	10^{-30}	10^{-14}
$MPF128$	0.95	31.57	217.9	10^{-38}	10^{-35}
$MPF256$	0.98	32.4	220.24	10^{-76}	10^{-75}

We emphasize here that most of the numerical computation timing is spent for getting a numerical certificate not for performing the actual computation of the roots!

3.4 Polynomial optimization

In the following examples, the border basis computation is combined with Semi-Definite Programming to compute the optimum of a polynomial function over a basic semi-algebraic set [2].

The minimizer ideal is zero-dimensional and the algorithm outputs a numerical approximation of the minimizer points and generators of the minimizer ideal,

after a finite number of relaxations. The SDP solver used in this computation is MOSEK.

The table records the number of variables (v), the number of inequality and equality constraints (c), the maximum degree of the constraints and of the polynomial to minimize (d), the number of minimizer points (sol), the maximal order (o), the maximal number of parameters (p), the maximal size of the moment matrices (s) in the SDP problems, and the total CPU time in seconds (t).

problem	v	c	d	sol	o	p	s	t
⋄ Robinson	2	0	6	8	4	21	15	0.10
⋄ Motzkin	2	0	6	4	4	26	15	0.08
⋄ Motzkin perturbed	3	1	6	1	5	167	56	0.90
⋄ L'01, Ex. 1	2	0	4	1	2	8	6	0.022
⋄ L'01, Ex. 2	2	0	4	1	2	8	6	0.022
⋄ L'01, Ex. 3	2	0	6	4	4	25	15	0.075
L'01, Ex. 5	2	3	2	3	2	14	6	0.037
F, Ex. 4.1.4	1	2	4	2	2	4	3	0.023
F, Ex. 4.1.6	1	2	6	2	3	6	4	0.023
F, Ex. 4.1.7	1	2	4	1	2	4	3	0.022
F, Ex. 4.1.8	2	5	4	1	2	13	6	0.031
F, Ex. 4.1.9	2	6	4	1	4	44	15	0.11
F, Ex. 2.1.1	5	11	2	1	3	461	56	4.61
F, Ex. 2.1.2	6	13	2	1	2	209	26	0.46
F, Ex. 2.1.3	13	35	2	1	2	2379	78	34.55
F, Ex. 2.1.4	6	15	2	1	2	209	26	0.43
F, Ex. 2.1.5	10	31	2	1	2	1000	66	12.31
F, Ex. 2.1.6	10	25	2	1	2	1000	66	6.05
F, Ex. 2.1.7(1)	20	30	2	1	2	10625	231	1083.60
F, Ex. 2.1.7(5)	20	30	2	1	2	10625	231	1117.33
F, Ex. 2.1.8	24	58	2	1	2	3875	136	311.54
F, Ex. 2.1.9	10	11	2	1	2	714	44	1.98
F, Ex. 3.1.3	6	16	2	1	2	209	26	0.61
L'09 cbms1	3	3	3	5	3	26	17	0.14
L'09 rediff3	3	3	2	2	2	7	7	0.06
L'09 quadfor2	4	12	4	2	3	48	19	0.45
Simplex	15	16	2	1	2	3059	120	65.73
Tensor Ex. 4.2	6	0	8	4	8	2340	210	59.38

The examples L'09 are from [13], L'01 from [15] and F are from [10]. New equality constraints are added, following [1], to the initial problems in the examples marked with ⋄. The example "Simplex" is the optimization of a quadratic polynomial over the simplex. "Tensor" is an example from best rank-2 approximation of a tensor from [19]. Experiments are made on an Intel Core i5 2.40GHz with 8Gb of RAM.

References

1. M. Abril Bucero and B. Mourrain. Exact relaxation for polynomial optimization on semi-algebraic sets. http://hal.inria.fr/hal-00846977, 2013.

2. M. Abril Bucero and B. Mourrain. Border Basis relaxation for polynomial optimization. *J. Symb. Comp.*, 74:378–399, 2015.
3. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling an A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992. `http://www.netlib.org/lapack/`.
4. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symb. Comp.*, 24(3-4):235–265, 1997.
5. R.M. Corless, P.M. Gianni, and B.M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In W.W. Küchlin, editor, *Proc. ISSAC*, pages 133–140, 1997.
6. W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 4-0-2 — A computer algebra system for polynomial computations. `www.singular.uni-kl.de`, 2015.
7. J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, J.W. H. Liu, and Xiaoye S. Li. A supernodal approach to sparse partial pivoting. In *SIAM J. Matrix Anal. Appl.* 20:720–755, 1999.
8. J.-C. Faugère. Fgb: A library for computing gröbner bases. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *ICMS*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87. Springer, 2010.
9. K. Fujisawa and M. Fukuda and K. Kobayashi and M. Kojima and K. Nakata and M. Nakata and M. Yamashita, SDPA (SemiDefinite Programming Algorithm), 2008.
10. C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gumus, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. *Handbook of Test Problems in Local and Global Optimization.* Kluwer Academic Publishers., 1999.
11. S. Graillat and Ph. Trébuchet. A new algorithm for computing certified numerical approximations of the roots of a zero-dimensional system. In *ISSAC 2009*, pages 167–173, 2009.
12. L. Huot. *Polynomial systems solving and elliptic curve cryptography.* PhD thesis, Université Pierre et Marie Curie (UPMC), 2013.
13. J.-B. Lasserre. *Moments, positive polynomials and their applications.* Imperial College Press, 2009.
14. J.-B. Lasserre, M. Laurent, B. Mourrain, P. Rostalski, and P. Trébuchet. Moment matrices, border bases and real radical computation. *J. Symb. Comp.*, 2012.
15. J.B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11:796–817, 2001.
16. B. Mourrain and P. Trébuchet. Generalized normal forms and polynomials system solving. In M. Kauers, editor, *ISSAC 2005*, pages 253–260, 2005.
17. B. Mourrain and Ph. Trébuchet. Border basis representation of a general quotient algebra. In J. van der Hoeven, editor, *ISSAC 2012*, pages 265–272, 2012.
18. MOSEK ApS. The MOSEK optimization library. `www.mosek.com`, 2015.
19. G. Ottaviani, P-J. Spaenlehauer, and B. Sturmfels. Exact solutions in structured low-rank approximation. *SIAM J. Matrix Anal. Appl.* 35(4):1521–1542, 2014.
20. B. Parisse. Giac/XCas, a free computer algebra system, 2008. Technical report, University of Grenoble.
21. Ph. Trébuchet. A new certified numerical algorithm for solving polynomial systems. In *SCAN2010*, pages 1–8, 2010.